An Expert-free evaluation of Scientific Summary through Diagram

Quang Vu
quangvu@gatech.edu

***Abstract*—**This paper proposes a solution to evaluate summary of a scientific article through diagram analysis. The model diagram used for evaluation is constructed solely base on the reading text, and does not require extra input from human experts. By fully automate this evaluation process, it helps bring a tight feedback loop that will help students to have a positive reinforcement of the knowledge of the article.

Video demo

1 INTRODUCTION

It has been showed that writing summary improves the synthesized knowledge from the students on what they read (Graham and Hebert, 2010). However, students would not get feedback on their writing summary, whether it is too short, too long, or too identical to the original text. Also, evaluating such summaries requires the grader to also read the article and develop their own summary before they can do evaluation. It adds up the turn-around time before students can be get feedback on the material they synthesized.

Also, empirical study showed that concept mapping and diagram, not only more efficient in information searching and recognizing, but also improved the thought process on student before they started writing (Harrell, 2008). However, traditional teaching method favors evaluating writing summary over diagram evaluating, due to the inconsistency of interpret those diagrams.

This project provides a tool that aims to assist the students in self-evaluating their diagram summary of the article they read, by using technology to address 2 tasks: auto extract summary information from reading text, and rating score to help validating their understand. The project hopes that by using the tools, it helps students to reinforce the information they collect from their reading.

2 RELATED WORK

(Novak, 1991) proposed learning by concept map on science subjects for first and second grade students. Since then, multiple researches have followed up on that approach by improving the process and the scoring model for evaluating concept map from student (Chang et al., 2005) (Gouli et al., 2005) (Wu et al., 2012). Their solutions all require an evaluation from human experts in order to compare with diagram from students. The referenced diagrams are also built by one or many human experts.

Automatic summarization is done with two main approaches: extraction and abstraction (S. Gupta and S. K. Gupta, 2019). Extraction is the process of pick out important keywords from the article, while abstraction is synthesizing the keywords in the article into more cohesive words. There are many different solutions with regard to both approaches: n-grams frequency analysis (Hovy et al., 2006), important content ranking analysis (Saggion et al., 2002), informativeness ranking (Filippova, 2010). Those solutions explored algorithm approach to identify key information.

Recently with the rise of neural network and deep learning, more works have been done on automated, data-base approach on finding information. (Crossley et al., 2019), (Maharjan et al., 2018) and (Ruseti et al., 2018), (Nallapati et al., 2016, August 26) use recurrent neural network (RNN) and long-short-term-memory (LSTM) network for summarizing and for scoring the summary. For the goal of this project, which is diagram extraction, there are several approaches, that are similar to our goal, with graph-base analysis (Ganesan et al., 2010, August), and abstract meaning presentation (AMR) (Banarescu et al., 2013), which mainly focuses on the relationship between entities. On the direction of AMR, work from (Foland and Martin, 2017, July) shows parsing of semantic content through graph spanning. (Konstas et al., 2017, July) and (S. Zhang et al., 2019) both show that AMR can be very effective with RNN/LSTM. Zhang (2019) achieves state of the art for populart LDC2014/LDC2017 trainset for AMR parsing. (Lyu and Titov, 2018) applies the aligner to transform back and forth AMR and sentence sequence.

In very recent years, Transformer (Vaswani et al., 2017) (Turc et al., 2019) and BERT (Devlin et al., 2019) have been the breakthroughs in natural language processing (NLP) and language model, showing multiple improvements in previ-

ous tasks. Thus, many recent researches apply those models for their problems: extreme summary of article in one sentence (Cachola et al., 2020, May 2), summary of scientific document through reviewer's comments (Liu, 2019), citation identification and summarization (Zerva et al., 2020, May 7), length controllable summarization (Saito et al., 2020).

3 DESIGN & IMPLEMENTATION

3.1 Tasks Definition

We break the goal of the project into multiple tasks, that will lead our design and implementation:

- Interface to take input from user.
- Extract diagram from article:
  - Extract main concepts: from the article, get `k`-words with importance-score for summarization.
  - Build relationship between concepts: build the relationship among `k`-words.
- Compare and score input diagram and extracted diagram.

3.2 Architecture Overview

3.2.1 *User Interface (UI) module*

The UI module will allow user to create and edit their diagram. It is built as a web app with HTML and `d3` Javascript library.

3.2.2 *Diagram module*

This module is designed as a backend task, where the extraction from the reading text serves as the expert diagram for evaluating. The key objective of this module is to identify main concepts, extract them, and decomposing them into smaller concepts represent in the article. The output information from this module should be as condense as possible. Chunk of text, sentences, paragraph should be represented as descriptive words.

3.2.3 *Scoring module*

The scoring module, which is implemented in Python backend of the web app, will handle comparison between diagram from student and diagram extracted from the reading text. It will determine how much score will be given for each

component/sub-component in the diagram.

3.3 Implementation

3.3.1 *User Interface (UI) module*

Creating diagram aims to be very straightforward and easy to use, to avoid interfere with their thought process. Two methods are provided for user to create their diagram: through YAML input file, and add/remove node one by one.

The YAML input file is provided to help user load their noted diagram, while the manual editing interface is provided to help simulating their interaction with the visual model, as they would normally with pencil and paper. The format for YAML file is available in Appendix.

3.3.2 *Diagram module*

Extract Important Words—

The goal of this task is to have a set of words with their importance score in regard to the referenced article. We begin the task by looking at each sentence in text, and try to determine the importance score of the sentence, by considering whether it will be included in the summary and whether it contains special keywords such as "explain", "finding", etc... With each sentence have its score, we assign each token (ie: word in the sentence) its token score. The sentence score is equally divided to all its sentence score. For example:

```
sentence: A--B--C--D
sentence_score: 1.0
token_score: A = 0.25, B = 0.25, C = 0.25, D = 0.25
```

The list of tokens in the sentence is filtered to exclude stop words ("the", "a", "an", "of", "in"...), non-important part of speech (adverb, article, pronounce...), so that the score is given to concepts, entities and their relationship in the sentence and is not wasted on trivial words that do not contribute to the summarization. As of part of this step, we also normalize verb tense (ie: go, goes, went...) as it is the feature that does not contribute much to our goal of summarization.

As an implementation note, we choose to represent a token node with these properties:

```
Node
    * name
    * score
    * part of speech type
    * sentence index
    * adjacent neighbors
    * root
```

When we establish the neighborhood/adjacency relationship between nodes in the sentence, the relationship is pair-wise across all nodes:

```
sentence:    A--B--C
A.neighbors:  {B, C}
B.neighbors:  {A, C}
C.neighbors:  {A, B}
```

It should also be noted that with this kind of bi-directional relationship, our graph is undirected graph, which we will need to handle later when try to convert to the tree diagram.

Collapse/Merge Node—

After we collect words and their score in each sentence, we proceed on merging them base on their similarity. If two tokens are similar enough, we merge them into one node, along with their connection to neighbor nodes. As an implementation note, the merging is done with `UnionFind` algorithm to speed up the processing runtime. The high level pseudo-code to illustrate this step is available in Appendix.

Intuitively, this merging means we establish a relationship between words from sentences that are far away from each other in the article. It also means words that appear with higher frequency will merge more than other words, and thus have higher score.

Controllable Length—

The desired length that we want for our target summary is controlled with 2 parameters: the level of similarity to merge words, and the number of words we collect. Lower threshold of similarity means more words will dissolve into

other words, while lower number of collected words means more words will get ignored (thus filter out the noise).

This approach for controlling lenght through number of collected words is developed independently, but we are later aware that it is very similar with the approach from (Saito et al., 2020) for controllable length. The small difference between our approach and their approach is we are using percentage of captured words, while theirs sets a specific length for the number of words to be considered.

Build Relationship Tree—

The main goal of this task is to break our undirected graph into a directed, acyclic graph (ie: tree). The directed graph is formed by traversing our nodes in the order of higher score, and remove the opposite direction (ie: the path of lower score node node to its parent, a higher score node). The pseudo-code for this task is available in Appendix

Our method of tracking neighbor nodes and merging nodes allows the same node to be referenced by different other nodes. In order to form a tree representation, we let node to be duplicated into different branches (node re-entrancy) (Banarescu et al., 2013) (Konstas et al., 2017, July) (Lyu and Titov, 2018).

```
Before:
        /--A--\
    root---B---C
After:
        /--A---C1
    root---B---C2
(where C1 and C2 are the same node)
```

3.3.3 *Scoring module*

The algorithm for calculating the score is taken from (Wu et al., 2012) and (Chang et al., 2005), which fundamentally takes the ratio of intersecting neighbors over union of neighbors between nodes in two graph. Formally, if we have NA, NB which are the set of neighbor nodes of node A from graph 1, and node B from graph 2, then the match score between A and B is:

$$score = \frac{|NA \cap NB|}{|NA \cup NB|}$$

We should note that in Wu's paper, the score takes into consideration the matching of its children nodes. So, if one of the children of node A results in partial match, A will also be marked for partial match. For our purpose of scoring summary, we choose to ignore this consideration, with the rationale to avoid double counting of the mismatch. This decision makes our calculation closer to the proposed algorithm from Chang's paper.

As mentioned in the *Diagram module*, nodes can be non-unique across the graph. Thus, in our implementation, we choose to identify the node by its combination of depth, parent node, and its ID, instead of just its ID. In another words, we identify nodes by their position in the graph. That identification works under the constraint that no duplicated node under the same parent, which is ensured by our diagram module, but is not implemented for the UI module under the scope of this project.

In the algorithm from referenced papers, nodes from the input diagram, that are not found in the solution diagram, will have a score of `0.0`. This wrong answer deduct the total score by reduce the match ratio from its neighbor nodes. For our purpose in this project, we also we introduce a small penalty point `-0.01` for nodes that are not found in the solution diagram. For example:

```
Solution diagram:  A-->B
Answer diagram:    A-->B-->C-->D
```

With the original algorithm, there will be no score difference between answer of `A-->B-->C` versus `A-->B-->C-->D`. With our modification, each node C and D will lower the total score by `-0.01`. The goal of this design is to discourage the user to provide answer that is more detail than necessary to the solution diagram.

4 RESULT

Our approach successfully capture some of the key concept, however the result of our diagram extraction falls much lower than our expectation. When compares to human evaluation, the matching rate is about 10%. Moreover, the graph

constructed on single word barely represent good semantic meaning. Some node in the diagram need to be concatenate to be a longer phrase to represent better meaning.

Our approach only performs well with small amount of sentences, approximately 10 or less. For perspective, average scientific paper is around 3,500 words, approximately 400 sentences. When the text get longer, the diagram picks up many unnecessary details, and the diagram gets filled with high frequency but very generic words: "article", "improvement", "result"...

5 DISCUSSION

One of our researches in the early stage of the project (S. Gupta and S. K. Gupta, 2019) points out `word2vec` as one of the most popular tools for NLP tasks, among many other tools. That leads our decision to use `word2vec` as the main language model for this project. Unfortunately, `word2vec` has several drawback that heavily affect the implementation and the performance of the project.

Lack of vocabulary. The Google News Word2Vec model that we use has 3 millions words in its vocabulary. However, that is not enough to cover many words that we would need for our summary task, especially since we are summarize scientific documents, which will have many domain-specific words. This limitation heavily affects our process of capturing and merging words. If the word is not in the vocabulary we will not be able to find its similarity with other words, thus we have to exclude it. Many important keywords could have been ignored by this limitation.

Lack of features. The Google News Word2Vec has 300 features. However, we are convinced that those number of features are not enough for the distinguishing of similarity we would need for our project. As example, Word2Vec similarity for `"car"` and `"vehicle"` is `0.7821`, while the similarity for `"represent"` and `"representation"` is `0.4137`. Such range of difference is difficult to determine an applicable threshold for similar words, while word-similarity is an important factor in our project.

Micro grammar. One of the main reasons we choose to summarize article as diagram, instead of popular method of sentence summary, is the removal of linguistic features that are not contribute much into the overall meaning (verb tense, article, adjective/adverb...). With our approach, we successfully remove

those features at document level, paragraph level, and sentence level. However, by filtering out the grammar rule too far, we face a problem with reconstructing our word tokens into meaningful phrases. As example, the phrase "abstract meaning representation" should be kept together as one token instead of three separated tokens "abstract", "mean", "representation", or another example that the phrase "part of speech" would not be able to reconstruct meaningfully if we filter out the article "of".

We consider solving this problem with taking into account group of words. However, not only that will significantly increase the computation cost, with the limitation of Word2Vec vocabulary it would be infeasible. Instead of that, we are more interested in the approach that BERT model used for question/answer task: use marker to mark the beginning and the end of the phrase, then use those markers to output the phrase "as is" in original text. For recap on this point, the disconnection of phrase pushes us to re-think about the level where we want remove and where we want to keep the linguistic features in the original text.

Acronym. One of the corner cases that come up is the translation of acronym: "recurrent neural network" <-> RNN. Whenever the acronym is used, it is the signal that it will likely be a frequently used label thorough the article. However, with the lack of vocabulary, it is almost certainly that all the acronyms will not get into the summary keywords.

Dataset. At the beginning of the project, we are aware of the advantage of deep learning solution over manual feature-engineering our extraction. We try to find the appropriate dataset that can help us with extracting and determining the score for each word. We are very interested in applying the approach with AMR, due to its focus on abstract relationship between entities, that relates to diagram constructing. However, both dataset LDC2014/LDC2017, that are very popular for ARM task, are not publicly available.

During our research for the project, we learn that applying either extractive summary or abstractive summary will depend heavily on the data that we will use to train our model. Dataset with abstractive summary will require more effort to architect the network to yield score related keywords, due to the unclear word-to-word relationship between the train data and the target. In another hand, extractive summary will be more domain specific. News data such as CNN Daily News and Daily Mall do not contain related keywords to help train the model

for scientific articles.

Near the end of our project, we discover SciSummnet dataset (Yasunaga, R. Zhang, et al., 2017) (Yasunaga, Kasai, et al., 2019, July 17) (Chandrasekaran et al., 2019) that is closest for our purpose. The dataset contains both gold summary and cited references that include score for gold summary consideration. Unfortunately, we did not have enough time left in the project to investigate further on using this dataset.

Evaluation metric. ROUGE-metric (Lin, 2004, July) is a very popular method for measuring quality of the summarization task, and it is used as the scoring metric in almost all the papers that we review. However, its most popular Python implementation,`rouge-score`, has several limitations: not filtering out stop words, requires exact match instead of fuzzy match or similarity match. The required exact match makes it tricky for us to have a good measurement, considering we already have difficulty with phrase. If the gold summary contains "meaning", it would return no match if our result has "mean", which is the result of our clean up.

Furthermore, the overlap matching method in ROUGE has more advantage for measuring sequence-sequence matching, but for the goal of building diagram like our project, we would have lower precision score due to filtering linguistic words. In another words, it is more challenging for us to get good measurement for constructing diagram with the gold summary in dataset being summary sentences.

BERT. Near the end of our project, we know more about BERT and think it would be a very promising improvement for our task. Not only BERT has better context awareness and larger range of feature columns (768 vs 300 in Word2Vec), it also has more extensible way of handling language modeling. A word in BERT is not directly tied up with a vector like Word2Vec, making it easier to fine-tune BERT to have a vector representation of new word, phrase, or sentence.

The implementation of BERT also includes built-in tokenizer that is built on `WordPiece`, that would help tokenize special and domain-specific words easier. For example, `"tensorflow"` is tokenized into `["tensor", "##flow"]`. This method of tokenizing not only would make reconstructing words easier, but also would mitigate the exact-match issue that we have with `rouge_score`. If anyone want to follow up on the direction of this project, we would highly rec-

ommend invest more time with BERT for future research.

6 CONCLUSION

While we are convinced that our diagram tool will improve students' experience in retrieve and recall the information they acquire from their reading, as well as we think it will help with their self-learning process, the measure of precision for our approach is much lower than what we expect. We think the removal of linguistic features will help student focus better and recall better on the key concepts that they want to retain. However, our implementation has several drawback that heavily affect the benefit that this project can bring.

During the development of this project, we have much clearer vision on which area need to be improved, in order to improve the performance of our tool. We are able to pinpoint and analyze the issues that draw down our performance. We also learn about new tools, new breakthrough in NLP research, as well as found appropriate dataset that are promising for future development. We hope this project can be a good reference resource for future development.

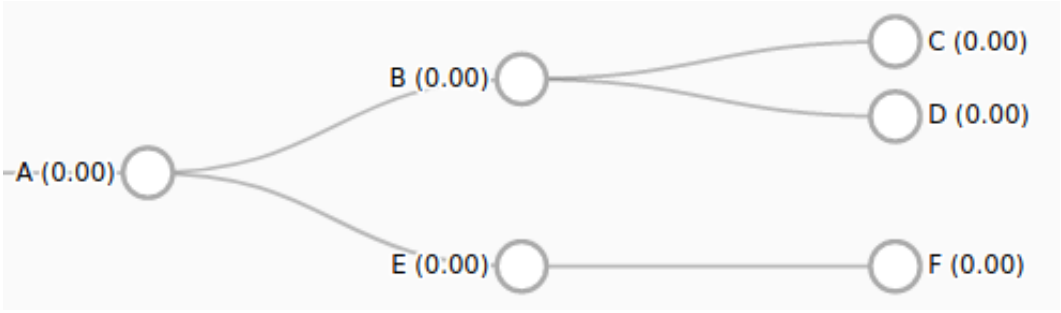
7 APPENDIX

7.1 YAML input

Format of the `YAML` file for diagram input:

```yaml
A:
    B:
        C: ""
        D: ""
    E:
        F: ""
```

Would output the diagram:



7.2 Merge Node pseudo-code

```
L := the ordered list of nodes.
For each node u in L:
    For each node v in L, such that v is not u:
        If v.score > u.score, swap u-v.
        If u and v is similar enough, merge them:
            u := UnionMerge(u, v)
            u.score := sum of (u.score, v.score)
            u.neighbors := union of (u.neighbors, v.neighbors)
```

7.3 Convert Graph to Tree pseudo-code

```
L := list of nodes ordered by score
S := empty list
For each node u in L, that u not in S:
    Add u to S
    For each v in u.neighbors:
        Remove path v -> u
```

REFERENCES

1. Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., ... Schneider, N. (2013). Abstract meaning representation for sembanking, 9.
2. Cachola, I., Lo, K., Cohan, A., & Weld, D. S. (2020, May 2). TLDR: Extreme summarization of scientific documents. *arXiv:2004.15011 [cs]*. arXiv: 2004.15011. Retrieved June 26, 2020, from http://arxiv.org/abs/2004.15011
3. Chandrasekaran, M. K., Yasunaga, M., Radev, D., Freitag, D., & Kan, M.-Y. (2019). Overview and results: CL-SciSumm shared task 2019. In *In proceedings*

of joint workshop on bibliometric-enhanced information retrieval and NLP for digital libraries (BIRNDL 2019). Retrieved from https://github.com/WING-NUS/scisumm-corpus

4. Chang, K.-E., Sung, Y.-T., Chang, R.-B., & Lin, S.-C. (2005). A new assessment for computer-based concept mapping. *Educational Technology & Society*, *8*(3), 138–148. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.6230
5. Crossley, S. A., Kim, M., Allen, L., & McNamara, D. (2019). Automated summarization evaluation (ASE) using natural language processing tools. In S. Isotani, E. Millán, A. Ogan, P. Hastings, B. McLaren, & R. Luckin (Eds.), *Artificial intelligence in education* (pp. 84–95). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-030-23204-7_8
6. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805 [cs]*. arXiv: 1810.04805. Retrieved June 25, 2020, from http://arxiv.org/abs/1810.04805
7. Filippova, K. (2010). Multi-sentence compression: Finding shortest paths in word graphs, 9.
8. Foland, W. & Martin, J. H. (2017, July). Abstract meaning representation parsing using LSTM recurrent neural networks. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 463–472). ACL 2017. Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1043
9. Ganesan, K., Zhai, C., & Han, J. (2010, August). Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics (coling 2010)* (pp. 340–348). Beijing, China: Coling 2010 Organizing Committee. Retrieved from https://www.aclweb.org/anthology/C10-1039
10. Gouli, E., Gogoulou, A., Papanikolaou, K., & Grigoriadou, M. (2005). Evaluating learner's knowledge level on concept mapping tasks. In *Fifth IEEE international conference on advanced learning technologies (ICALT'05)* (pp. 424–428). Fifth IEEE international conference on advanced learning technologies (ICALT'05). Kaohsiung, Taiwan: IEEE. doi:10.1109/ICALT.2005.143
11. Graham, S. & Hebert, M. (2010). *Writing to read: Evidence for how writing can improve reading*. Alliance for Excellent Education Washington, DC.

12. Gupta, S. & Gupta, S. K. (2019). Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, *121*, 49–65. doi:10.1016/j.eswa.2018.12.011
13. Harrell, M. (2008). No computer program required: Even pencil-and-paper argument mapping improves critical-thinking skills. *Teaching philosophy*, *31*, 351. doi:10.5840/teachphil200831437
14. Hovy, E., Lin, C.-Y., Zhou, L., & Fukumoto, J. (2006). Automated summarization evaluation with basic elements, 4.
15. Konstas, I., Iyer, S., Yatskar, M., Choi, Y., & Zettlemoyer, L. (2017, July). Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 146–157). ACL 2017. Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1014
16. Lin, C.-Y. (2004, July). ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics. Retrieved July 15, 2020, from https://www.aclweb.org/anthology/W04-1013
17. Liu, Y. (2019). Fine-tune BERT for extractive summarization. *arXiv:1903.10318 [cs]*. arXiv: 1903.10318. Retrieved July 26, 2020, from http://arxiv.org/abs/1903.10318
18. Lyu, C. & Titov, I. (2018). AMR parsing as graph prediction with latent alignment. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 397–407). Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers). Melbourne, Australia: Association for Computational Linguistics. doi:10.18653/v1/P18-1037
19. Maharjan, N., Gautam, D., & Rus, V. (2018). Assessing free student answers in tutorial dialogues using LSTM models. In C. Penstein Rosé, R. Martínez-Maldonado, H. U. Hoppe, R. Luckin, M. Mavrikis, K. Porayska-Pomsta, ... B. du Boulay (Eds.), *Artificial intelligence in education* (pp. 193–198). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-93846-2_35
20. Nallapati, R., Zhou, B., santos, C. N. d., Gulcehre, C., & Xiang, B. (2016, August 26). Abstractive text summarization using sequence-to-sequence RNNs and beyond. *arXiv:1602.06023 [cs]*. arXiv: 1602.06023. Retrieved June 27, 2020, from http://arxiv.org/abs/1602.06023

21. Novak, J. D. (1991). A twelve-year longitudinal study of science concept learning. *American Educational Research Journal, Spring, 1991, 28*, 117–153. Retrieved from https://www.jstor.org/stable/pdf/1162881.pdf
22. Ruseti, S., Dascalu, M., Johnson, A. M., McNamara, D. S., Balyan, R., McCarthy, K. S., & Trausan-Matu, S. (2018). Scoring summaries using recurrent neural networks. In R. Nkambou, R. Azevedo, & J. Vassileva (Eds.), *Intelligent tutoring systems* (pp. 191–201). Lecture Notes in Computer Science. Cham: Springer International Publishing. doi:10.1007/978-3-319-91464-0_19
23. Saggion, H., Teufel, S., Radev, D., & Lam, W. (2002). Meta-evaluation of summaries in a cross-lingual environment using content-based metrics. In *Proceedings of the 19th international conference on computational linguistics-volume 1* (pp. 1–7). tex.organization: Association for Computational Linguistics.
24. Saito, I., Nishida, K., Nishida, K., Otsuka, A., Asano, H., Tomita, J., ... Matsumoto, Y. (2020). Length-controllable abstractive summarization by guiding with summary prototype. *arXiv:2001.07331 [cs]*. arXiv: 2001.07331. Retrieved July 14, 2020, from http://arxiv.org/abs/2001.07331
25. Turc, I., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Well-read students learn better: On the importance of pre-training compact models. *arXiv:1908.08962 [cs]*. arXiv: 1908.08962. Retrieved July 18, 2020, from http://arxiv.org/abs/1908.08962
26. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv:1706.03762 [cs]*. arXiv: 1706.03762. Retrieved July 16, 2020, from http://arxiv.org/abs/1706.03762
27. Wu, P.-H., Hwang, G.-J., Milrad, M., Ke, H.-R., & Huang, Y.-M. (2012). An innovative concept map approach for improving students' learning performance with an instant feedback mechanism. *British Journal of Educational Technology*, *43*(2), 217–232. https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8535.2010.01167.x. doi:10.1111/j.1467-8535.2010.01167.x
28. Yasunaga, M., Kasai, J., Zhang, R., Fabbri, A. R., Li, I., Friedman, D., & Radev, D. R. (2019, July 17). ScisummNet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 7386–7393. doi:10.1609/aaai.v33i01.33017386
29. Yasunaga, M., Zhang, R., Meelu, K., Pareek, A., Srinivasan, K., & Radev, D. R. (2017). Graph-based neural multi-document summarization. In *Proceedings of CoNLL 2017*.

30. Zerva, C., Nghiem, M.-Q., Nguyen, N. T. H., & Ananiadou, S. (2020, May 7). Cited text span identification for scientific summarisation using pre-trained encoders. *Scientometrics*. doi:10.1007/s11192-020-03455-z
31. Zhang, S., Ma, X., Duh, K., & Van Durme, B. (2019). AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 80–94). Proceedings of the 57th annual meeting of the association for computational linguistics. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1009